Optimal assignment for carpooling - DRAFT

Irith Ben-Arroyo Hartman*

July 3, 2013

Abstract

Car pooling is a service which encourages citizens to share vehicles while driving to common, or nearby destinations. The problem has been defined and approached from many points of view. We propose here a new, simple, integer linear programming formulation, as well as an equivalent graph theoretic formulation. We prove that this formulation of the problem is NP-hard, even in the case that every car allows at most two passengers, except for the driver. If, on the other hand, the set of drivers and cars are known, then the car pooling problem is tractable: we give an algorithmic proof and a characterization theorem in the spirit of Hall's theorem for matchings in bipartite graphs.

1 Introduction

Car pooling is a service which encourages citizens to share vehicles while driving to common, or nearby destinations. The advantages of carpooling are multiple: it decreases the number of cars on the roads, thus decreasing traffic congestion, vehicle emissions, and noise. It saves fuel, time, parking spaces, and decreases the number of accidents. In addition, it encourages sociability between people, and reduces stress in driving to work.

The car pooling problem has been approached from many, diverse, points of view, see [2, 4]. The main challenge is how to match between people to share a ride, and to decide who picks up whom with their vehicle, so that it will be worthwhile both for the driver and all the passengers. Even if we assume that every driver can pick up at most one other passenger, the problem is challenging. We propose here new, simple formulations of the problem, and prove that some are NP-complete, and others tractable.

In [7, 8] an automatic service for carpooling is described. People register their *periodic trip executions* (PTE), these are periodic trips, where the base period is typically one week. Each PTE contains information about the trip such as origin and destination of the trip, earliest and latest departure and arrival times, the maximal detour distance acceptable, and the capacity of the car (except for the driver), if it is available. In addition, each PTE also contains registration information of the person such as age, gender, educational level, special interests (like music type preferences), job category, driver license availability, etc. The system suggests to individuals to share a car based on the informations of the their trips, as well as their registration information. The assumption is that continued successful cooperation between people requires some level of similarity. After the system makes its suggestions, together with appropriate financial incentives for the driver, the individuals evaluate the proposed carpool, negotiate it, and possibly agree to cooperate. Once the individuals

*Caesarea Rothschild Institute, University of Haifa, Israel *e-mail*: irith.hartman@gmail.com

have shared a trip, they can evaluate each other (e.g. was the driver/passenger punctual? Did the driver obey traffic rules? Was the vehicle comfortable?) and give the feedback to the system, for consideration in future trips.

The model we use is a weighted directed graph, as in [7]. Each vertex corresponds to a PTE, and x and y are joined by a directed edge (x, y) if and only if it is a feasible possibility that the owner of PTE x will execute the trip by riding in the vehicle used by the owner of PTE y. The weight of edge (x, y) is the estimated probability for the negotiation between x and y to succeed. This probability takes into account the information related to both PTE's x and y, as well registration information and feedbacks. In [7] an attempt was made to describe the carpooling problem as an Integer Linear Programming problem. We give in Section 2 a simpler, more elegant description of the problem. In Section 3 we formalize the problem as a graph theoretic problem, and in Subsection 3.1 prove it is NP-Hard. Finally, in Section 3.2 we define a tractable problem where the set of drivers and vehicles is known, and we would like to assign passengers to them. We give an algorithmic solution to it, as well as proving a characterization theorem.

2 Linear Programming Formulation

Let V denote the set of Periodic Trip Executions, or for short, trips. Each trip is associated with an individual who either drives his own vehicle, or travels as a passenger with some other driver. A directed edge $(i, j) \in E$ of weight w_{ij} corresponds to the compatibility of person i to get a ride with driver j (in driver j's vehicle), i.e. it represents the probability that the negotiation between them succeeds and i can travel in $j's$ vehicle. Each driver j also has a maximum capacity of people that he can take in his vehicle (not including himself!), denoted by c_j.

The carpooling problem can be described by an integer linear programming problem: We have a variable y_i for each $i \in V$ which equals one if driver i takes his own vehicle and zero otherwise. For each pair of vertices i and j, variable x_{ij} has value one if individual i has a ride in j's vehicle, and zero otherwise. The LP problem is stated below:

(LP)

$$\text{Maximize } \textstyle\sum_{ij} w_{ij} x_{ij}$$

Subject to

$$\begin{aligned}
\text{for all } i \in V,\ y_i + \sum_j x_{ij} &\leq 1 && (1)\\
\text{for all } i, j \in V,\ x_{ij} &\leq y_j && (2)\\
\text{for all } j \in V,\ \sum_i x_{ij} &\leq c_j y_j && (3)\\
\text{for all } i, j \in V,\ x_{ij} \in \{0,1\}, y_i \in \{0,1\} & && (4)
\end{aligned}$$

Constraint (1) ensures that each individual either drives his own vehicle, or gets a ride with at most one driver, but not both. Inequality (1) can be replaced by equality. Constraint (2) makes sure that individual j is marked as a driver if there exists any individual i who gets a ride with him. Constraint (3) bounds the number of passengers driver j takes with him, and (4) is the integrality constraints.

3 Graph-Theoretic Formulation

3.1 The General "Hard" Problem

We introduce some graph theoretic concepts in order to describe the problem above as a graph theoretic problem.

Definition 3.1 (Star, Directed Star, Star Family) *A* star *is a graph ($K_{1,t}$) consisting of a center vertex called* root *and vertices connected to it called* leaves*. A* directed star *is a star whose edges are all directed towards the root of the star. Denote the set of vertices and edges of a star S by $V(S)$ and $E(S)$, respectively. A* star family *is a collection $\Gamma = \{S_r\}$ of vertex disjoint directed stars. We denote the set of edges and vertices covered by a star family Γ by $E[\Gamma]$ and $V[\Gamma]$, respectively. Denote by $R[\Gamma]$ ($F[\Gamma]$) the set of all roots (leaves) of the stars in Γ, respectively.*

Definition 3.2 (Feasible Star family) *Given a graph $G = (V, E)$ and let $c : V \to \mathbb{N}$. A star family $\Gamma = \{S_r\}$, $r \in R$ is* feasible *if each star with root r has in-degree at most $c(r)$.*

Definition 3.3 (Star Partition) *Given a directed graph $G = (V, E)$, a* star partition *in G is a star family that covers $V(G)$. I.e it is a collection of vertex disjoint directed stars that cover $V(G)$.*

We remark that a directed star may have in-degree zero, in which case it is a single vertex $v = K_1$ which we call the *trivial star*. Therefore, every graph has a feasible star partition, the trivial one, consisting of $|V|$ trivial stars.

The LP above is equivalent to the following problem:

Problem 3.1 *Let $G = (V, E)$ be a directed graph with edge weights $w : E \to \mathbb{R}$, and let $c : V \to \mathbb{N}$. Find a feasible star partition $\Gamma = \{S_r\}$, $r \in R$ such that the total weight of the edges of the stars is maximized.*

Solution to problem 3.1 guarantees that as many people as possible are happy with their priorities for "taking rides."

Problem 3.1 does not guarantee that the number of vehicles used is as small as possible. It is easy to construct an example where the solution to the problem stated above does not find a minimum number of vehicles, see figure [fig1]. However, when $w_{ij} = 0$ or 1 these problems are equivalent, as is seen in the claim below:

Claim 3.1 *If the edge weight function w_{ij} is either 0 or 1, then the LP above is equivalent to the problem of finding star partition with minimum number of stars (i.e. a minimum number of vehicles).*

Proof: If the weight function w_{ij} is either 0 or 1 then an optimal solution to the LP above finds the maximum number of edges in a star partition of G. For any covering of $V(G)$ with d disjoint directed stars, the total number of edges in the stars is $|V| - d$, since each star contains one less edge than the number of vertices covered by it. This implies that the minimum number of stars covering the graph equals $|V|$ - Max $\sum_{ij} w_{ij}x_{ij}$. Hence, an optimal solution to the LP corresponds to a minimum feasible star partition, and vice versa.

■

From the claim above it follows that when w_{ij} is either 0 or 1 then Problem 3.1 is equivalent to the following problem:

Problem 3.2 *Let $G = (V, E)$ be a directed graph, and let $c : V \to \mathbb{N}$. Find a feasible star partition $\Gamma = \{S_r\}$, $r \in R$ containing a minimum number of stars.*

We will show that Problem 3.2 is NP-Hard. This implies that Problem 3.1 is also NP-Hard since Problem 3.2 is a special case of it. We will reduce the Minimum Dominating Set Problem (MDSP) to Problem 3.2. A *dominating set* for a graph $G = (V, E)$ is a subset S of V such that every vertex not in S is adjacent to at least one member of S. The domination number $\gamma(G)$ is the number of vertices in a smallest dominating set for G. The dominating set problem concerns testing whether $\gamma(G) \leq K$ for a given graph G and input K. It was shown to be NP-complete in [5].

Theorem 3.1 *Problem 3.2 is NP-Hard.*

Proof: Given an instance of MDSP consisting of a graph G and input K, we transform it to an instance of Problem 3.2. We create from G a symmetric directed graph G' where for each $(u, v) \in E(G)$ we have (u, v) and (v, u) in $E(G')$. Let $c : V \to \mathbb{N}$ be defined by $c(v) = \text{Maxdeg}$ where Maxdeg is the maximum degree in G. It is not difficult to see that every dominating set S for G corresponds to a star cover in G' where the centres of the stars are in S. By removing some edges of the stars it is possible to get a star partition which is feasible by the choice of the function c. Conversely, every feasible star partition $\Gamma = \{S_r\}$, $r \in R$ in G' corresponds to a dominating set S in G, where $S = R$. Hence $\gamma(G) \leq K$ if and only if G' has a feasible star partition with at most K stars. ■

In the proof above we transformed the Minimum Dominating Set Problem to a special case of Problem 3.2 where the capacity function c is, $c(v) = \text{Maxdeg}$ for all $v \in V$, meaning that, in fact, there is no bound on the degrees of the stars. We will show that even if the degree of every star is small, the problem is still NP-Hard.

Theorem 3.2 *Problem 3.2 is NP-Hard even for the special case where $c : V \to \mathbb{N}$ is defined by $c(v) \leq 2$.*

Proof: We will reduce the problem of partitioning into paths of length two (PPL2) to our problem. In the PPL2 problem we are given a graph $G = (V, E)$ with $|V| = 3q$ for some positive integer q. We want to determine if there is a partition of V into q disjoint sets $V_1, V_2, \ldots, V_q$ of three vertices each so that each set V_i spans a path of length two or a cycle of length three. This problem is NP-complete using a reduction from the 3-dimensional matching, see [5]. Given an instance of PPL2 consisting of a graph $G = (V, E)$ with $|V| = 3q$, we transform it to an instance of Problem 3.2 with $c(v) \leq 2$. As in the previous proof, we create from G a symmetric directed graph G' with capacity function $c(v) = 2$ for all $v \in V$. Now, every star partition in G' corresponds to a partition of V in G into disjoint paths of length 2, and vice versa, thus proving the theorem. ■

We have shown that if we want minimize the number of cars by assigning passengers to cars, even if the capacity of every car is at most two (except for the driver), the problem is NP-hard. We remark that if $c(v) = 1$ for all $v \in V$ (i.e. every car has room for one passenger except for the driver) then Problem 3.2 is equivalent to the problem of finding maximum weight matching in a graph which has an efficient algorithm by J. Edmonds, see [11].

3.2 Some Tractable Problems

Consider now the following easier problem than Problem 3.2, where the roots of the stars are known in advance, and the problem consists of finding a feasible star partition with the given roots. The analogy to the car pooling problem is that the drivers and cars are known, and the problem consists of assigning passengers to the cars.

Problem 3.3 *Let $G = (V, E)$ be a directed graph, $c : V \to \mathbb{N}$, and let $R \subseteq V$ be a set of potential roots. Does there exist a feasible star partition $\Gamma = \{S_r\}$, where $r \in R$?*

In other words, given a set of potential drivers and cars, can we assign all individuals to the cars, without violating the capacity constraints of every car? A solution of Problem 3.3 will imply a solution to the following problem:

Problem 3.4 *Let $G = (V, E)$ be a directed graph, let $c : V \to \mathbb{N}$, and let $\Gamma = \{S_r\}$, $r \in R$ be a feasible star partition. Does there exist a feasible star partition $\Gamma' = \{S'_r\}$, $r \in R'$ such that $R' \subset R$?*

Another way of phrasing Problem 3.4 is, given a set of drivers with their vehicles and corresponding passengers, can we 'get rid' of some of the vehicles by rearranging all the passengers in some subset of the given vehicles?

To solve Problem 3.4, assuming we have a solution to Problem 3.3, let $\Gamma = \{S_r\}$, $r \in R$ be a feasible star partition. We delete from Γ some star, say S_i, with root r_i. Let $R' = R \setminus \{r_i\}$. If there exits a feasible star partition $\Gamma' = \{S'_r\}$, $r \in R'$ for some $1 \leq i \leq |R|$ then we are done. Otherwise, the solution to Problem 3.4 is negative.

We will describe a polynomial algorithm for problem 3.3. Given R, the algorithm begins with some feasible star family $\Gamma = \{S_r\}$, $r \in R$. (An example of such a family is the family consisting of trivial stars centered in vertices in R). We then pick some vertex x not covered by Γ and try to extend Γ to include x as well. If we succeed, we continue extending Γ to include other uncovered vertices. Otherwise, there exists no feasible star partition $\Gamma = \{S_r\}$, where $r \in R$.

We begin with some definitions:

Definition 3.4 (exposed, saturated, unsaturated) *Let $G = (V, E)$ be a directed graph, $c : V \to \mathbb{N}$, and let $\Gamma = \{S_r\}$, $r \in R$ be a feasible star family. A vertex which is not covered by any of the stars in Γ is called an* exposed vertex. *A root vertex $r \in R$ is* Γ-saturated *(or, for short,* saturated *) if the degree of the star S_r equals $c(r)$. Otherwise, it is said to be* Γ-unsaturated *(or, for short,* unsaturated).

Definition 3.5 (Undirected Path, Forward and Backward edges) *Let $G = (V, E)$ be a directed graph. An* undirected path *P in G is a sequence $P = (v_0, e_1, v_1, e_2, v_2, ..., e_l, v_l)$ such that, for each $1 \leq i \leq l$, either $e_i = (v_{i-1}, v_i) \in E(G)$ or $e_i = (v_i, v_{i-1}) \in E(G)$, and all the vertices are distinct. We assign a direction to P from v_0 to v_l , so that if $e_i = (v_{i-1}, v_i)$ then e_i is considered a* forward *edge, and if $e_i = (v_i, v_{i-1})$ then e_i is considered a* backward *edge. For brevity, we also denote P by the sequence of its vertices, $P = (v_0, v_1, \ldots, v_l)$.*

Definition 3.6 (Alternating, Augmenting Path) *Given a graph $G = (V, E)$, $c : V \to \mathbb{N}$ and a feasible star family $\Gamma = \{S_r\}$, $r \in R$. A path in G is* $\Gamma-$alternating *if its edges alternate between edges in $E[\Gamma]$ and edges not in $E[\Gamma]$, where the edges in Γ are backward edges, and the edges not in Γ are forward edges.*

A Γ- *alternating path* P *is* Γ-augmenting *if it is of the form* $xr_1f_1r_2f_2\dots r_kf_ku$ *where* x *is an exposed vertex, for each* $i \in [1,k]$, r_i *is a saturated root,* f_i *is a leaf adjacent to* r_i *and* u *is an unsaturated root. All edges* $(x,r_1),(f_1,r_2),(f_2,r_3),\dots,(f_k,u)$ *are forward edges, and the rest are backward edges.*

Lemma 3.3 *Let* $\Gamma = \{S_r\}$, $r \in R$ *be a feasible star family in a graph* $G = (V,E)$, *and let* $x \in V \backslash V[\Gamma]$ *be an exposed vertex. If* G *contains a* Γ- *augmenting path with origin at* x, *then there exists a feasible star family* $\Gamma' = \{S'_r\}$, $r \in R$, *where* $V[\Gamma'] = V[\Gamma] \cup \{x\}$.

Proof: Let $P = (xr_1f_1r_2f_2\dots r_kf_ku)$ be a Γ-augmenting path with origin at x. It is not difficult to see that if we take the symmetric difference between $E(P)$ and $E[\Gamma]$, i.e. $E(P)\Delta E[\Gamma] = (E(P) \cup E[\Gamma]) \setminus (E(P) \cap E[\Gamma])$ we get the edge set of another feasible star family $\Gamma' = \{S'_r\}$, with the same set of roots $r \in R$, but which covers an additional vertex, x. ■

The converse to Lemma 3.3 is less trivial, but it also holds. Before proving it we give an algorithm for searching for a Γ-augmenting path with origin x.

We first define a Γ-alternating tree, similar to an M-alternating tree used in the *Hungarian Method* (see [1]). Let x be an exposed vertex, i.e. $x \notin V[\Gamma]$. A tree $T \subseteq G$ is called a Γ-*alternating tree rooted at* x if (i) $x \in V(T)$ and (ii) for every vertex v of T, the unique (x,v)-path in T is a Γ-alternating path. The search for a Γ-augmenting path with origin x involves 'growing' a maximal Γ-alternating tree T rooted at x. Let $F' = V(T) \cap F[\Gamma]$ and $R' = V(T) \cap R[\Gamma]$ denote the set of leaves and roots of Γ in T, respectively. Initially, T consists of just the single vertex x. It is then grown in such a way that, at any stage, either

(i) every root vertex in T is a Γ-saturated root of some star in Γ, and all of its leaves are on T (see Fig Tree-a), or

(ii) T contains a Γ-unsaturated root vertex (see Fig Tree-b)

If (i) is the case then at every stage of growing the tree we have $N^+(\{x\} \cup F') \supseteq R'$. If $N^+(\{x\} \cup F') \supset R'$ then there is an $r \in R[\Gamma] \backslash R'$ adjacent to a vertex $v \in \{x\} \cup F'$. If r is Γ-saturated then the in-degree of the star rooted in r is exactly $c(r)$. Let $v_1, v_2, \dots, v_{c(r)}$ be the leaves of the star rooted in r. We grow T by adding the vertices r and $v_1, v_2, \dots, v_{c(r)}$ and the edges (v,r) and $(v_1,r),(v_2,r),\dots,(v_{c(r)},r)$. We are then back in case (i). If r is Γ-unsaturated, we grow T by adding the vertex r and the edge (v,r) resulting in case (ii), where the unique (x,r)-path of T is a Γ-augmenting path with origin x, as required.

The complexity of growing a Γ-alternating tree rooted at x is $O(|V| + |E|)$ since a breadth first search algorithm is sufficient to grow it.

Lemma 3.4 *Let* $\Gamma = \{S_r\}$, $r \in R$ *be a feasible star family in a graph* $G = (V,E)$, *and let* $x \in V \backslash V[\Gamma]$ *be an exposed vertex. If* G *contains no* Γ-*augmenting path with origin at* x, *then there exists no feasible star family* $\Gamma' = \{S'_r\}$, $r \in R$, *where* $V[\Gamma'] \supseteq V[\Gamma] \cup \{x\}$.

Proof: We grow a maximal Γ-alternating tree T rooted at x. Since G contains no Γ-augmenting path with origin at x, case (i) above holds and $N^+(\{x\} \cup F') = R'$. Furthermore, all the vertices in R' are saturated roots. Hence, $|F'| = \sum_{r \in R'} c(r)$, which is the maximum number of leaves that the set R' of roots can have in a feasible star family. Hence, not all vertices of $\{x\} \cup F'$ can be covered by the set R' of roots. Since $N^+(\{x\} \cup F') = R'$, other root vertices in $R \setminus R'$ are not relevant in covering $\{x\} \cup F'$. Hence no feasible star family with roots in R can cover $\{x\} \cup F' \subseteq V[\Gamma] \cup \{x\}$. This proves the Lemma. ■

From Lemmas 3.3 and 3.4 we immediately get:

Theorem 3.5 *Let $\Gamma = \{S_r\}$, $r \in R$ be a feasible star family in a graph $G = (V, E)$, and let $x \in V \setminus V[\Gamma]$ be an exposed vertex. Then there exists a feasible star family $\Gamma' = \{S'_r\}$, $r \in R$, where $V[\Gamma'] = V[\Gamma] \cup \{x\}$ if and only if G contains a Γ-augmenting path with origin at x.*

As a corollary, we get a "Hall type" characterization of graphs with a given star partition whose roots are a given subset of V, similar to Hall's theorem for the existence of a perfect matching in a bipartite graph [6].

Corollary 3.6 *Let $G = (V, E)$ be a directed graph, $c : V \to \mathbb{N}$ and let $R \subseteq V$. Then there exists a star partition $\Gamma = \{S_r\}$ where $r \in R$, if and only if for each subset $F \subseteq V \setminus R$, $\sum_{r \in N^+(F) \cap R} c(r) \geq |F|$.*

Proof: The 'only if' part is easy. If, by contradiction, there exists a subset F of leaves with $|F| > \sum_{r \in N^+(F) \cap R} c(r)$, then we do not have enough roots, adjacent to F, to cover all the vertices in F. To prove the 'if' part, let Γ be a star family with roots in R, which covers maximum number of vertices. If there exists some exposed vertex $x \in V \setminus V[\Gamma]$, then by the maximality of Γ there exists no Γ-augmenting path with origin in x, and by the proof of Lemma 3.4, a set $\{x\} \cup F'$ is found with $N^+(\{x\} \cup F') = R'$ and $|\{x\} \cup F'| > \sum_{r \in R'} c(r)$, contradicting the condition in the corollary. ■

We remark that Corollary 3.6 can be derived from the Ore-Ryser Theorem [10] and Tutte's f-factor Theorem [12]. To see this we construct the bipartite graph $G' = (V_1 \cup V_2; E')$ where $V_1 = V \setminus R, V_2 = R$ and $E' = \{(u, v) | u \in V_1, v \in V_2, (u, v) \in E\}$. Problem 3.3 is equivalent to the problem of finding a maximum f-factor in G' where the function f is defined $0 \leq f(v) \leq 1$ for all $v \in V_1$, and $1 \leq f(v) \leq c(v)$ for all $v \in V_2$. This problem can be solved also by constructing an auxiliary bipartite graph G'_f as in the proof of the f-factor Theorem by Tutte [12].

Consider now a simpler (tractable) case of Problem 3.1.

Problem 3.5 *Let $G = (V, E)$ be a directed graph with edge weights $w : E \to \mathbb{R}$, let $c : V \to \mathbb{N}$, and let $R \subseteq V$ be a set of potential roots. Find a feasible star family $\Gamma = \{S_r\}$, $r \in R$ such that the total weight of the edges of the stars is maximized.*

Problem 3.5 is equivalent to the problem of finding a maximum weight f-factor in the associated bipartite graph G' as defined above. It can be solved by constructing an auxiliary bipartite graph G'_f as in the proof of the f-factor Theorem by Tutte [12], and finding a maximum weight matching in G'_f.

References

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory*, Springer, 2008.

[2] R. Baldacci, V. Maniezzo, and A. Mingozzi, *An Exact Method for the Car Pooling Problem Based on Lagrangian Column Generation*, Operations Research vol. 52, No 3, 422-439, 2004.

[3] G. Even, N. Garg, J. Konemann, R. Ravi, and A. Sinha, *Covering Graphs using Trees and Stars*, Algorithms and Techniques, 2003.

[4] E. Ferrari, R.Manzini, A. Pareschi, A. Persona, and A. Regattieri, *The Car Pooling Problem: Heuristic Algorithms Based on Savings Functions*, Journal of Advanced Transportation, Vol. 37 No.3, 243-272 (2003).

[5] M.Gary and D. Johnson *Computers and Intractability: A Guide to the Theory of NP-Completness* , W.H. Freeman, San Francisco, 1979.

[6] P. Hall, *On Representatives of Subsets* , J. London Math. Soc. 10 (1): 263, 1935

[7] L. Knapen, D. Keren, A. Yasar, S. Cho, T. Bellemans, D. Janssens and G. Wets, *Estimating scalability issues while finding an optimal assignment for carpooling*, The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)

[8] L. Knapen, A. Yasar, S. Cho, D. Keren, A. Abu Dbai, T. Bellemans, D. Janssens, G. Wets, A. Schuster, I. Sharfman, and K. Bhaduri, *Exploring Graph-theoretic Tools for Matching in Carpooling Applications*, Journal of Ambient Intelligence and Humanized Computing.

[9] W. Lin and P. Che-Bor Lam, *Star Matching and Distance Two Labelling*, Taiwanese Journal of Mathematics, Vol. 13, No. 1, pp. 211-224, February 2009.

[10] O. Ore, *Graphs and subgraphs*, Trans. Amer. Math. Soc., 84, 109-136, 1957.

[11] A. Schrijver, Combinatorial Optimization, Polyhedra and Efficiency, Volume A, Springer, 2003

[12] ...